Evolving Spatial and Frequency Selection Filters for Brain-Computer Interfaces

Ricardo Aler, Inés M. Galván, and José M. Valls

Abstract— **Machine Learning techniques are routinely applied to Brain Computer Interfaces in order to learn a classifier for a particular user. However, research has shown that classification techniques perform better if the EEG signal is previously preprocessed to provide high quality attributes to the classifier. Spatial and frequency-selection filters can be applied for this purpose. In this paper, we propose to automatically optimize these filters by means of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). The technique has been tested on data from the BCI-III competition, because both raw and manually filtered datasets were supplied, allowing to compare them. Results show that the CMA-ES is able to obtain higher accuracies than the datasets preprocessed by manually tuned filters.**

I. Introduction

The aim of EEG-based Brain-Computer Interfaces (BCI) is to detect patterns in user EEG signals in order to control a computer or an external device [1], [13], [6], [10]. As an example, patterns corresponding to motor imagery (the user imagines that one of his body parts is moving), object rotation imagery or thinking of words, can be recognized in the EEG signal. The high variability of EEG patterns among different subjects makes machine learning classification techniques the tool of choice [3]. Thus, classifiers can be trained from user-generated data by means of supervised machine learning techniques [4]. The classifier can then be used to detect patterns on real-time EEG data.

It is very difficult to learn classifiers from the raw EEG data for two reasons. First, the number of attributes is too large and second, it is known that patterns are best detected on the frequency-domain rather than in the time-domain of the original raw data. Therefore, the raw EEG signal is usually preprocessed before the training stage. Three kinds of transformations are commonly used [4]: spatial filters, the Fourier Transform (to convert to the frequency domain), and band-pass filters. Spatial filters are useful because the signal detected at one electrode can come from different parts of the brain. Spatial filters can generate a more localized signal for every electrode. For instance, a common approach is to subtract the average of surrounding electrodes from each electrode, thus highlighting the signal generated just below that particular electrode. Once the signal has been spatially filtered, it can be transformed to the frequency domain by means of the Fast Fourier Transform. Moreover, it is known that the patterns of interest are located in particular frequency bands. As an example, motor imagery is accompanied by increase/decrease of amplitude in the frequency band from 8Hz to 15Hz and can be easily observed in the frequency-domain signal (this phenomena is called event-related desynchronization (ERD) and event-related synchronization (ERS) [16]). Therefore, it is important to use attributes from the frequency domain. But in order to reduce the number of attributes supplied to the classifier, it is also crucial to select the right frequency-bands, bearing in mind that different users will display ERD/ERS phenomena on slightly different bands. Frequency-selection filters are used in this paper for that purpose.

An appropriate preprocessing of the signal is acknowledged to be very important in order to get high classification accuracy. Filters are typically adjusted by hand, following a process of trial and error. This requires some experience on building and adjusting filters that at the end, may turn out not to be optimal. There are also some approaches to compute filters automatically. Most work have been carried out on computing spatial filters by common spatial patterns (CSP) [5], [4]. This approach allows to determine data projections, represented by linear transformations, that maximize the variance of signals of one class and minimize the variance of signals of the other class. There are some extensions to CSP that allow to also learn band-pass (or spectral) filters [11], [2], [17]. Maximizing the variance of one class while minimizing that of the other clearly increases the discriminability of the two classes, so it is expected that most classifiers will obtain high accuracies on the filtered data. But CSP and its variants do not directly optimize the accuracy of the classifier on the data.

In this paper, we propose to optimize simultaneously both spatial and frequency-selection filters by means of Evolution Strategies (particularly, the CMA-ES algorithm [14], [9]). Evolution Strategies are able to optimize functions without making strong assumptions about it. Therefore, given a classifier, our approach tries to find the optimal filters that maximize the accuracy of the algorithm, instead of using surrogate measures (like CSP does) to determine the separability of the classes.

This paper is structured as follows. Section II explains how to learn a classifier from filtered data. Section III describes how filters can be evolved by CMA-ES. The approach is empirically tested in Section IV. Finally, Section V summa-

Ricardo Aler is with the Department of Computer Sciences, Universidad Carlos III de Madrid, Avenida Universidad 30, 28911 Leganés, Spain; email: aler@inf.uc3m.es).

Inés M. Galván is with the Department of Computer Sciences, Universidad Carlos III de Madrid, Avenida Universidad 30, 28911 Leganés, Spain; email: igalvan@inf.uc3m.es).

José M. Valls is with the Department of Computer Sciences, Universidad Carlos III de Madrid, Avenida Universidad 30, 28911 Leganés, Spain; email: jvalls@inf.uc3m.es).

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

rizes what has been achieved and proposes new avenues of research.

II. Learning a classifier from filtered raw EEG data: Spatial and Frequency-selection filters

A m-class classifier is a function $C : \mathbb{R}^n \rightarrow \{c_1, c_2, \ldots, c_m\}$ that assigns a class c_i to an instance $(a_1, \ldots, a_n) \in \mathbb{R}^n$. The a_i are the instance's attributes. Classifiers can be learned from a set of labeled instances called the training data. Labeled data contain an additional attribute indicating the class of the instance: $(a_1, \ldots, a_n, c)$. Before raw EEG data can be used as training instances it has to be preprocessed by computing the instances's attributes from the raw signal. This Section will explain in detail how to get from raw EEG to the classifier, by filtering the acquired data with spatial and frequency-selection filters.

Learning a classifier can be carried out in three steps:

1) Acquiring the raw EEG data for training purposes
2) Preprocessing the raw data in order to create the attributes for the training instances. Spatial and frequency-selection filters are employed
3) Training the classifier by means of supervised learning techniques

A. Data acquisition

In order to train a classifier, an acquisition session has to be carried out to capture data from the user. We will describe now the setting for the type of acquisition sessions used in this paper. This setting corresponds to what is usually called "classification of continuous EEG without trial structure" in the literature.

The EEG signal is recorded from c electrodes (or channels) located at different places on the scalp of the user. The continuous signal at the electrodes is discretized with sampling frequency f. If the acquisition session lasts for s seconds, then a temporal series of $f * s$ data points (or time instants) will be generated for each of the c channels. Therefore, a session can be represented as a $(f*s) \times c$ matrix, that will be named $\mathbf{Session}$. A session is made of periods. In each period, the user is instructed to achieve a particular mental state. Examples of mental states are imagining left-hand movement or imagining right-hand movement. Thus, every time instant will be associated with a particular mental state. The sequence of mental states in a session can be represented by a $(f * s) \times 1$ column vector. It will be named the $\mathbf{Class}$ vector, because it actually contains the classes required by the supervised learning method that will be used later.

B. Generating the set of training instances

It is from the $\mathbf{Session}$ matrix that the training data for the classifier has to be generated. The training set will be made of many training instances. Each training instance, denoted by I_n, is computed from a submatrix S extracted from the $\mathbf{Session}$ matrix[1]. In fact, S is a window that spans for t time instants. Thus, the first training instance will be obtained from a submatrix $S_1 = \mathbf{Session}(1 : t, 1 : c)$, the next training instance from $S_2 = \mathbf{Session}(1 + \delta t : t + \delta t, 1 : c)$, and so on. The nth instance will be generated from $S_n = \mathbf{Session}(1 + (n-1) * \delta t : t + (n-1) * \delta t, 1 : c)$. It is important to remark that if S_n is located in the transition between two mental states (two different thoughts), then the training instance is discarded in order to avoid noisy instances. However, this processing is performed only when generating the training set but but not for the testing instances, because in the later case, it is not known when the mental state changes.

Notation $(1 : t, 1 : c)$ means that the submatrix contains all time instants from 1 to t and all channels from 1 to c. This is a typical Matlab notation. So, each instance is obtained from a window S that moves in steps of size δt. If $\delta t = t$, then there is no overlap between the windows, otherwise there will be some overlap. Supervised training techniques require that all instances have been labeled. The class corresponding to S_n is stored in $\mathbf{Class}(t + (n-1) * \delta t)$.

In summary, the algorithm to generate the set of instances is:

Algorithm 1 Generation of the Training Instances

```
1: begin ← 1
2: end ← t
3: n ← 1
4: while Session has not been processed do
5:    S_n ← Session(begin : end, 1 : c)
6:    I_n ← GenerateInstance(S_n)
7:    TrainingSet ← TrainingSet ⋃ (I_n, Class(end))
8:    begin ← begin + δt
9:    end ← end + δt
10:   n ← n + 1
11: end while
12: Return(TrainingSet)
```

The procedure $\mathbf{GenerateInstance}$ will be described in the next Section.

C. Generating a Single Training Instance: applying spatial and frequency-selection filters

Now, the process to generate a training instance I_n from submatrix S_n will be described. It has three steps:

1) Apply the spatial filter L (a linear transformation of the training data)
2) Apply the Fast Fourier Transform (FFT)
3) Apply the frequency-selection filter B

There are many kinds of spatial filters (Common Average Referencing, Laplace Filtering, PCA, ICA, etc.), but most of them can be represented by a linear transformation on the original data. Therefore, our spatial filters are $c \times c'$ matrices, denoted by L, and the result of filtering is just the matrix product shown in Eq. 1

$$S'_n = S_n * L \tag{1}$$

[1] S stands for "signal".

The filtered matrix S'_n is a $t \times c'$ matrix. If $c' = c$, then S'_n has the same dimensions as the original matrix S_n. If $c' < c$, then the number of channels of S'_n is reduced. In some sense, the spatial filter L transforms c channels into c' channels.

The application of the Fast Fourier Transform to the spatially filtered data is given by Eq. 2.

$$S''_n = |FFT(S_n * L)| \tag{2}$$

where operator $|\ |$ computes the modulus of each one of the components of the matrix resulting from FFT. FFT returns complex numbers, with phase and modulus, but most research on BCI deal with the modulus only [4], therefore the phase will be ignored here. S''_n in Eq. 2 is also a $t \times c'$ matrix, but now the rows of the matrix belong to the frequency domain.

After the FFT transformation, rows from 1 to $t/2$ represent the frequency band $(0Hz, (f/2)Hz)$ (where f is the sampling frequency), with a resolution of $\delta f = (f/t)Hz$. The frequency bands contained in the matrix are $(0, \delta f)$, $(\delta f, 2 * \delta f)$, etc. Sometimes, the resolution δt is smaller than necessary, so it is convenient to work with wider frequency-bands. This can be achieved by averaging several consecutive frequency-bands.

A training instance can then be constructed by flattening S''_n and concatenating all its components as shown in Eq. 3.

$$\begin{aligned} I_n \quad = \quad & (S''_n(1,1), \ldots, S''_n(t/2,1), S''_n(1,2), \ldots, S''_n(t/2,2), \\ & \ldots, S''_n(1,c'), \ldots, S''_n(t/2,c')) \end{aligned} \tag{3}$$

where $S''_n(i,j)$ represents frequency-band $i \in \{1 \ldots t/2\}$ at channel $j \in \{1 \ldots c'\}$. This results in a large number of attributes $(t/2 * c')$, with many of them containing no physiological information. Classification problems with many irrelevant attributes usually lead to overfitting.

Frequency-selection filters can be learned to select the most appropriate frequency-bands for every user. In the classification context this amounts to attribute selection. In this paper, the frequency-selection filter is represented by a binary vector shown in Eq. 4.

$$B = (b_{1,1}, \ldots, b_{t/2,1}, b_{1,2}, \ldots, b_{t/2,2}, \ldots, b_{1,c'}, \ldots, b_{t/2,c'}) \tag{4}$$

where $b_{i,j} = 1$ means that the attribute that represents frequency band i and channel j is to be selected, otherwise to be removed.

We can summarize the process of creating training instance I_n from S_n as:

$$I_n = B(\mathbf{flatten}(|\mathbf{FFT}(S_n * L)|)) \tag{5}$$

where **flatten** is the flattening of the matrix (concatenating all of its components) and B is the application of the frequency-selection filter.

Figure 1 shows the basic steps of the instance generation procedure as explained before.

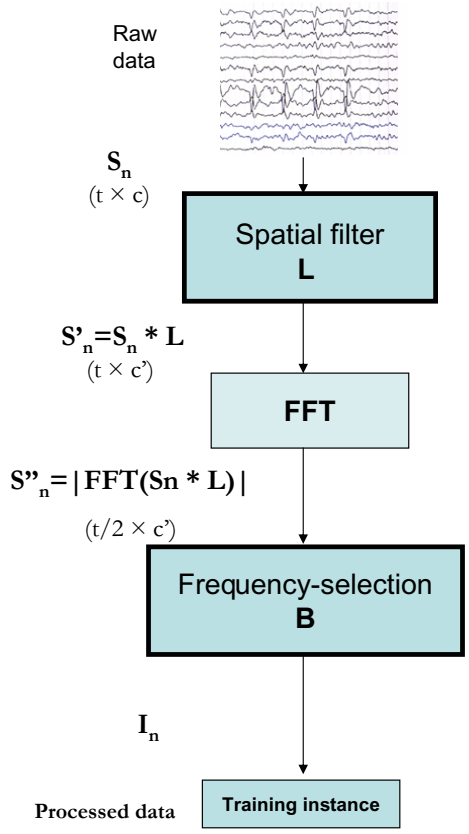

Fig. 1. Instance generation procedure

D. Training the classifier

Once the training set is available, many supervised classification techniques can be used to train classifier C: Support Vector Machines, Neural Networks, etc. Preliminary experiments showed that linear classifiers worked better than non-linear ones. We have chosen two linear classifiers: Linear Discriminant Analysis (LDA a.k.a. the Fisher Discriminant or FD) and linear-kernel Support Vector Machines (SVM). Both classifiers will be used in different contexts in the rest of the paper.

III. Evolution of Spatial and Frequency-selection Filters

The goal of this paper is to evolve spatial and frequency-selection filters that are optimal for some classifier C. From the previous sections, we have seen that in this work:

- Learning the spatial filter is equivalent to constructing a $t \times c'$ matrix L.
- Learning the frequency-selection filter is equivalent to constructing a binary vector B

In this Section, it will be explained how the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) has been used to optimize L and B for a FD classifier. CMA-ES (Covariance Matrix Adaptation Evolution Strategy) is one of the best evolutionary techniques for continuous optimization in difficult non-linear domains [14], [9]. CMA-ES is an Evolution Strategy where search is guided by a covariance matrix, which is adjusted and updated during the search process. CMA-ES works well in both local and global optimization tasks. An interesting feature of CMA-ES is that it requires almost no tuning of its parameters.

In order to specify the problem to CMA-ES, both the representation of candidate solutions (the chromosome) and the function to be optimized, must be described. In this work, CMA-ES is used to solve a minimization problem.

A. The chromosome

The chromosome contains the parameters to be optimized. In this case, these are the frequency-selection filter (B) and the spatial filter matrix L. CMA-ES evolves real numbers, so a flattened L matrix can be directly encoded in the chromosome. On the other hand, B contains binary values. They have been encoded as real numbers, but decoded as "0" if the integer part is odd and "1" if even.

B. The fitness function

In order to evaluate the quality of filters (B, L) encoded in a chromosome, a classifier C is built on the training data filtered by L and B, as explained in Section II. In this case, we have chosen C to be the Fisher Discriminant (FD). Next, C is applied on the same training data and the resulting classification error is computed. Although FD is linear, some overfitting was observed because the number of parameters in (B, L) to be adjusted by CMA-ES is high. To prevent overfitting, regularization measures were taken in order to minimize the number of components set to 1 in vector B in Eq. 4, which in turn determines the number of attributes of the training instances. This number has been normalized to $[0, 1]$ and is denoted by $|B|$. Eq. 6 displays the fitness function to be minimized.

$$\mathbf{fitness}(B, L) = \mathbf{Error} + \lambda |B| \tag{6}$$

where $\mathbf{Error} \in [0, 1]$ is the classification error on the training data and λ is the regularization parameter.

FD has been chosen because preliminary experiments have shown that linear classifiers are the best options for the data. As the fitness function is evaluated for every chromosome, its computation must be fast, and FD satisfies this requirement. Although linear-kernel SVM usually achieves higher accuracy, it is much slower. SVM can still be used at the end of the run, once the best (B, L) pair has been obtained. The rationale is that if a (B, L) pair works well for FD, similar or better results could be obtained with a better linear classifier (SVM). Of course, this will be tested experimentally.

IV. Experimental Validation

A. Data sets description

In this paper three datasets acquired in the IDIAP Research Institute will be used [12]. They have been previously tested in the 2005 BCI-III competition.[2] Each dataset contains data from a different subject during 4 non-feedback sessions. 32 electrodes were located on the subjects's scalp. There are 3 mental tasks, so this is a three-class classification problem:

- Imagination of repetitive self-paced left hand movements
- Imagination of repetitive self-paced right hand movements
- Generation of words beginning with the same random letter

All 4 sessions of a given subject were acquired on the same day, each lasting 4 minutes with 5-10 minutes breaks in between them. The subject performed a given task for about 15 seconds and then switched randomly to another task at the operator's request. EEG data is not splitted in trials since the subjects are continuously performing any of the mental tasks. Data was provided by the competition organizers in two ways: raw EEG signals with 32 electrodes, and data with precomputed features with 8 selected electrodes. In this paper, we use both versions of the datasets: our method to evolve filters is applied on the former while the later is used for comparison purposes.

The dataset with precomputed features provided by the competition organizers had been obtained by the following procedure. First, the raw EEG potentials were spatially filtered by means of a manually tuned surface Laplacian. Then, 16 times per second the power spectral density (PSD) in the band 8-32 Hz was estimated over the last second of data with a frequency resolution of 2 Hz. Additionally, physiological knowledge was used to select 8 centro-parietal channels (C3, Cz, C4, CP1, CP2, P3, Pz, and P4) out of the 32 original electrodes. As a result, an EEG sample is a 96-dimensional vector (8 channels times 12 frequency components).

B. Experimental Testing

As explained before, there are 4 sessions for each one of the three subjects. The three first sessions were available for training, while the last one was used for testing the classifiers designed by the participants of the competition. The parameters of our system have been adjusted to mimic the competition conditions. These are:

- Sampling frequency $f = 512$ Hz
- 1 second of data is used to construct every training instance, therefore $t = 512$
- Training instances are sampled 16 times per second, therefore $\delta t = \frac{512}{16} = 32$
- The number of electrodes is $c = 32$

Similarly to the competition precomputed features datasets, frequency bands outside the $(8, 32)$ Hz range have

[2] http://www.bbci.de/competition/iii/results/index.html#martigny.

been removed because they contain no physiological information. Also, the frequency-band width considered by the frequency-selection filter B has been set to 2Hz, as in the competition. This means, that the frequency-bands are $(8, 10)$ Hz, $(10, 12)$ Hz, ..., $(28, 30)$ Hz, $(30, 32)$ Hz. Therefore, B is made of 12 binary values. Our spatial filter L is a $(c \times c')$ matrix: it transforms c channels into c' channels. It is known that imagination of left (right) hand movements is related to a certain part of the right (left) hemisphere [4]. If we had only these two classes, it would be reasonable to set $c' = 2$. In this problem there is a third class (imagination of random words), therefore we have also tested $c' = 3$, finding that results are similar. Hence, c' has been set to 2 for the rest of experiments. Also, after some preliminary experiments λ has been set to 0.2.

In order to avoid overfitting, a stopping criterion that uses a validation set has been imposed on CMA-ES for all datasets. The validation set is obtained by mixing and randomizing the training instances from sessions 1, 2, and 3. 80% of data is selected for fitness computation and 20% for the validation-based stopping criterion. The validation classification error is measured at each iteration and evolution is stopped when the validation error becomes almost stable (more specifically, when it changes less than 0.005% for 30 iterations).

Next, results of the system run on every of the three subjects will be described. First, Figures 2,3 and 4 display a graphical evolution of the classification error on the training, validation, and the test sets for subject 1, 2 and, 3. It can be seen that evolution stops before the onset of overfitting. Every iteration in Figures 2,3 and 4 take approximately 3 minutes (in a computer with a 2.5Ghz CPU and 4Gb RAM).

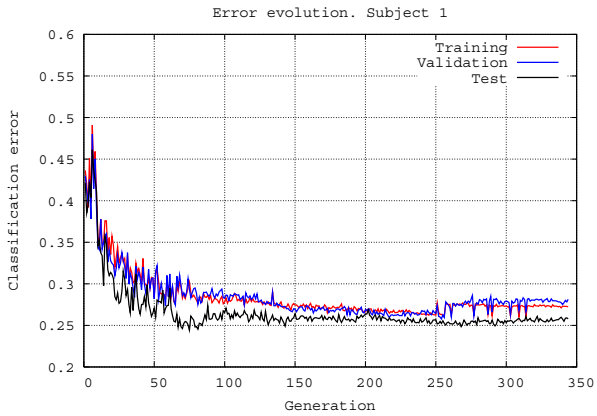

Fig. 2. Error evolution. Subject 1

Table I shows some details about the CMA-ES runs: the number of generations carried out, the number of frequency-bands selected by the best B filter, and the FD classification rate for the test data (session 4) for each subject. The Table also includes the test classification rate obtained with a linear SVM.[3] This SVM implementation solves multi-class problems by means of pairwise classification. SVM are

[3]Weka's SMO implementation has been used [8].

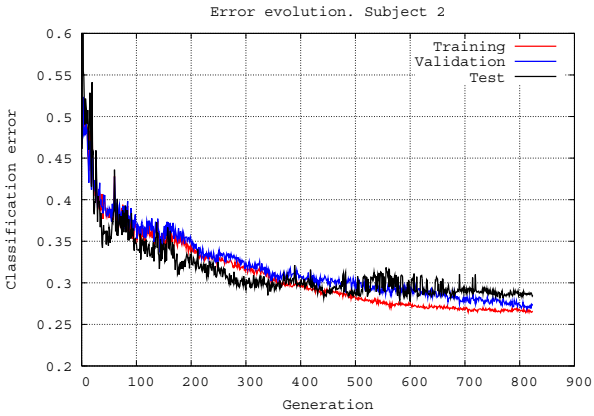

Fig. 3. Error evolution. Subject 2

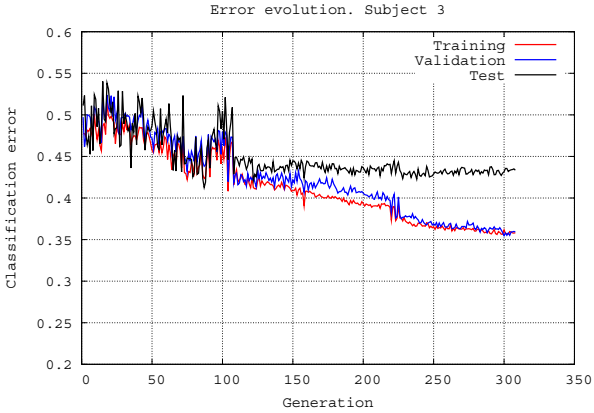

Fig. 4. Error evolution. Subject 3

generally better than FD because they return the maximal margin hyperplane which shows very good generalization capabilities. SVM have not been used for fitness computation during the evolutionary process because of the large computational cost involved: FD are much faster. But linear SVM can be used at the end of CMA-ES on the filtered data, once the best (B, L) pair has been found, with the aim of improving results further.

TABLE I

RESULTS OF FILTERS (B, L) EVOLVED BY CMA-ES: NUMBER OF GENERATIONS INVOLVED, NUMBER OF FREQUENCY-BANDS SELECTED, AND TESTING ACCURACY (PERCENT) BY FD AND SVM.

	Subject 1	Subject 2	Subject 3
Generations	344	824	308
Frequency-bands selected	3	2	2
Test Accuracy (FD)	74.20	71.52	56.65
Test Accuracy (SVM)	77.96	71.36	56.76

It can be observed in Table I that only a few frequency-bands have been selected (3 for subject 1 and 2 for subjects 2 and 3). Also, it can be seen that SVM provides better classification rates only for subject 1. For the rest of subjects, FD and SVM perform similarly.

First, the classification rates obtained by the evolved filters will be compared to the datasets with precomputed features supplied to the organizers of the competition. Let us remember that the later dataset was obtained by applying a carefully hand-adjusted surface Laplacian spatial filter and selecting the physiologically appropriate channels, as explained in Section IV-A. The features obtained by this manually tuned filter were high quality, as demonstrated during the competition: the best results were achieved by using these precomputed features rather than the raw signal dataset.[4] Table II compares the dataset filtered by the evolved filters with the precomputed features dataset. In both cases, SVM was applied to the filtered datasets (training was carried out with session 1, 2, and 3 and testing with session 4, for each subject). Given that the number of frequency-bands selected by the evolved filters is very small, we have also applied a WEKA attribute selection algorithm to the precomputed features dataset [8]. Different algorithms were tested and showed similar results. The selected algorithm uses a best-first search mechanism together with an information gain measure to evaluate sets of attributes [15]. It can be seen that results tend to improve when attributes are selected (at least, for subjects 1 and 2). In any case, results obtained by the evolved filters are better than the ones for precomputed features. This is true for all three subjects. For subject 1, the improvement is near to 3%. For subjects 2 and 3 the improvement is more that 5%.

TABLE II

COMPARISON BETWEEN THE CLASSIFICATION SUCCESS RATE USING PRECOMPUTED FEATURES DATA (WITH AND WITHOUT ATTRIBUTE SELECTION) AND RAW DATA FILTERED BY THE EVOLVED FILTERS.

	Subject 1	Subject 2	Subject 3
SVM and precomputed features	72.71	60.71	50.14
SVM and precomputed features + Attribute Selection	75.25 7	65.29 2	48.42 3
SVM and evolved filters Frequency-bands selected	77.96 3	71.36 2	56.76 2

Our results are also competitive with those obtained in the BCI-III Competition. One of the requirement of the competition was to provide the average of 8 consecutive samples (instances) in order to get a response every 0.5 seconds, because input vectors were computed 16 times per second. This smoothing process tends to improve results. Results shown in previous Tables I and II had been computed by considering all instances. In order to compare with competition results, all classification rates will be calculated by averaging every 8 consecutive instances. Table III displays these results. The first row shows the evolved filters results. The second row corresponds to the winner of the competition. Those results were been obtained from the precomputed dataset provided by the organizers but including extra capabilities such as detection transitions between mental states [7]. This helped to increase classification rates significantly. The third row shows the results obtained by Shiliang Sun[5] in the competition. These figures are relevant because they used the raw EEG data (just like our evolving filters approach does), instead of the precomputed data. They applied a multiclass Common Spatial Patterns filter, among other preprocessing tools and a SVM classifier.

TABLE III

COMPETITION SUCCESS RATE CLASSIFICATION

	Subject 1	Subject 2	Subject 3
Evolved Filters (SVM)	79.97	75.11	57.76
Competition Winner (preprocessed Data)	79.60	70.31	56.02
Best competition result on raw data	74.31	62.32	51.99

According to Table III, evolved filters are competitive with the winner of the competition (that used precomputed data). This is remarkable, given that it used a mental task transition detector in addition to preprocessing and classification algorithms. Also, evolved filters perform better than the best competition approach that worked on raw data.

V. CONCLUSIONS

This work has presented an evolutionary approach based on CMA-ES to create simultaneously both spatial and frequency-selection filters, in order to improve classification rates for brain-computer interfaces. This research started as a way to automatically adjust spatial filters, instead of having to follow a trial-and-error manual tuning process. By using spatial filters only, we obtained a large number of frequency-bands which are in turn the attributes for the classifier. Given that having a large number of attributes usually results in overfitting and that many of them are known to be physiologically irrelevant, the system evolves simultaneously the spatial filters and selects the most appropriate frequency-bands for each channel. The system allows to find both spatial and frequency-band selection filters adapted for every particular subject.

Results show that the evolved filters are better than those manually tuned. We have also compared our results with the winners of the BCI-III competition. On the one hand, our classification rates are similar to the winner who used the precomputed features and additionally a mental-task transition detector. We achieve similar results by only filtering the data. On the other hand, our results are significantly better than the best results in the competition that, as our system does, start from the raw EEG data.

ACKNOWLEDGMENT

This work has been funded by the Spanish Ministry of Science under contract TIN2008-06491-C04-03 (MSTAR project)

[4]See http://www.bbci.de/competition/iii/results/index.html#martigny. The first 8 best results used the precomputed features (column marked with PSD=Y). The best result obtained directly from the raw data comes only at 9th place (PSD=N))

[5]http://www.bbci.de/competition/iii/results/index.html#martigny

REFERENCES

[1] E.A. Curran and M.J. Stokes. Learning to control brain activity: a review of the production and control of eeg components for driving braincomputer interface (bci) systems. *Brain Cognition*, 51, 2003.

[2] G. Dornhege, B. Blankertz, M. Krauledat, F. Losch, G. Curio, and K. R. Muller. Combined optimization of spatial and temporal filters for improving braincomputer interfacing. *IEEE Transactions on Biomedical Engineering*, 53(11):2274–2281, 2006.

[3] Guido Dornhege, Matthias Krauledat, Klaus-Robert Muller, and Benjamin Blankertz. *Toward Brain-Computer Interfacing*, chapter General Signal Processing and Machine Learning Tools for BCI Analysis, pages 207–234. MIT Press, 2007.

[4] Guido Dornhege et al. (eds.). *Towards Brain-Computer Interfacing*, chapter General Signal Processing and Machine Learning Tools for BCI Analysis, pages 207–234. MIT Press, 2007.

[5] K. Fukunaga. *Introduction to statistical pattern recognition*. Academic Press, 1990.

[6] C. Neuper G. Pfurtscheller and N. Birbaumer. *Motor Cortex in Voluntary Movements*, chapter 14, pages 367–401. CRC Press, 2005.

[7] Ferran Galán, Francesc Oliva, and Joan Guardia. Using mental tasks transitions detection to improve spontaneous mental activity classification. *Medical and Biological Engineering and Computing*, 45(6):1741–0444, 2007.

[8] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[9] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[10] Andrea Kubler and Klaus-Robert Muller. *Toward Brain-Computer Interfacing*, chapter An Introduction to Brain-Computer Interfacing, pages 1–26. MIT Press, 2007.

[11] S. Lemm, B. Blankertz, G. Curio, and K. R. Muller. Spatio-spectral filters for improved classification of single trial eeg. *IEEE Transactions on Biomedical Engineering*, 52(9):1541–1548, 2005.

[12] J. del R. Millán. On the need for on-line learning in brain-computer interfaces. In *Proceedings of the International Joint Conference on Neural Networks*, Budapest, Hungary, July 2004. IDIAP-RR 03-30.

[13] Mourino J Millan J del R, Renkens F and W. Gerstner. Noninvasive brain-actuated control of a mobile robot by human eeg. *IEEE Trans Biomed Eng*, 51, 2004.

[14] A. Ostermeier, A. Gawelczyk, and N. Hansen. A derandomized approach to self-adaptation of evolution strategies. *Evolutionary Computation*, 4(2):369–380, 1994.

[15] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[16] G. Pfurtscheller and F. H. L. da Silva. Event-related synchronization of mu rhythm in the eeg over the cortical hand area in man. *NeuroScience Letters*, 174, 1994.

[17] R. Tomioka, G. Dornhege, G. Nolte, B. Blankertz, K. Aihara, and K. R. Muller. Spectrally weighted common spatial pattern algorithm for single trial eeg classification. Technical Report 40, Department of Mathematical Engineering, University of Tokyo, Japan, 2006.